Smoothing Splines

Contents

Introduction

Smoothing splines, also known as **penalized splines**, introduce a penalty term that controls the model's complexity and the smoothness of the function. This penalty term effectively balances the fit fidelity to the data with the simplicity of the spline function, reducing the risk of overfitting the model.

The formulation of a penalized spline can be expressed as the minimization of an objective function that includes both the squared error of the classic problem, as well as a penalty term, which depends on the second derivative of the spline function. Mathematically, this is represented as:

$$||y - X\beta||^2 + \lambda \int_0^1 (f''(x))^2 dx = \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \int_0^1 (f''(x))^2 dx. \tag{1}$$

The parameter λ is the **smoothing parameter** and the second derivative of f allows modeling its curvature. For values of λ tending to infinity, the estimation does not allow large fluctuations in the data, and f could be linear. On the other hand, lower values of λ give less weight to the penalty, resulting in more flexible models, but with a higher risk of overfitting. In the extreme case where $\lambda = 0$, the spline is unpenalized.

Since f is linear in the parameters β_i, it is possible to express the penalty as a quadratic form of β,

$$\int_0^1 (f''(x))^2 dx = \beta^T S \beta,$$

with S being a symmetric matrix of dimension $(q+1) \times (q+1)$. Each element of S located at position (i, j) is calculated through the integral $\int_0^1 b_i''(x) b_j''(x) dx$. Now, the minimization problem transforms into

$$\min_{\beta} ||y - X\beta||^2 + \lambda \beta^T S \beta. \tag{2}$$

The problem of estimating the degree of smoothness of the model is transformed into estimating the smoothing parameter λ. The goal is to find a λ value that ensures the approximation from the regression spline is as close as possible to f. There are multiple ways to optimize the value of λ, the most common strategy is based on cross-validation.

Cross Validation

Since the value of the function f describing the data is unknown, it is not possible to explicitly calculate the mean squared error between $f(x_i)$ and the approximation $\hat{f}(x_i)$. Consequently, an indirect approach is proposed that involves minimizing an estimation of the expected squared error, adjusted for variance.

Cross validation provides a practical method for estimating the performance of a model without needing the function f. Given the estimate of f, $\hat{f}$, the mean squared error is defined as:

$$M = \frac{1}{n}\sum_{i=1}^{n}(\hat{f}(x_i) - f(x_i))^2,$$

and the mean squared error of cross-validation,

$$V_0 = \frac{1}{n}\sum_{i=1}^{n}(\hat{f}^{[-i]}(x_i) - y_i)^2,$$

where $\hat{f}^{[-i]}$ is the model adjusted to all data except y_i.

Substituting the relationship $y_i = f(x_i) + \varepsilon_i$ into the definition of V_0 and considering that ε_i is independent of $\hat{f}^{[-i]}$ with zero mean, we have:

$$\mathrm{E}(V_0) = \mathrm{E}\left(\frac{1}{n}\sum_{i=1}^{n}(\hat{f}^{[-i]}(x_i) - f(x_i) - \varepsilon_i)^2\right)$$

$$= \mathrm{E}\left(\frac{1}{n}\sum_{i=1}^{n}\left((\hat{f}^{[-i]}(x_i) - f(x_i))^2 + 2\varepsilon_i(\hat{f}^{[-i]}(x_i) - f(x_i)) + \varepsilon_i^2\right)\right)$$

$$= \frac{1}{n}\,\mathrm{E}\left(\sum_{i=1}^{n}(\hat{f}^{[-i]}(x_i) - f(x_i))^2\right) + \sigma^2.$$

If $\hat{f}^{[-i]}$ is assumed to be a good approximation of $\hat{f}$, then $\mathrm{E}(V_0)$ becomes a reasonable approximation of $\mathrm{E}(M) + \sigma^2$. Therefore, minimizing V_0 emerges as a viable approach for minimizing M.

To determine the optimal value of λ in a penalized spline model, the following procedure is carried out:

A set of candidate values for λ is determined. For each chosen value of λ, the mean squared error of cross-validation, V_0, is calculated. The value of λ that minimizes V_0 among all calculated is selected. However, ordinary cross-validation, which involves fitting the model n times, can be computationally costly. Therefore, **generalized cross-validation** emerges as a computationally efficient alternative, effectively selecting the value of λ that optimizes the balance between fit and smoothness.

Generalized cross validation is expressed by the formula:

$$V_g = \frac{n\sum_{i=1}^{n}(y_i - \hat{f}_i)^2}{[\mathrm{tr}(I - A)]^2},$$

where A is the influence matrix of the model, $A = \mathbf{X}(\mathbf{X}^T\mathbf{X} + \lambda S)^{-1}\mathbf{X}$. This matrix allows evaluating how each data point influences the prediction, adjusting the error.

This approach provides an efficient tool for selecting the parameter λ, thus balancing the accuracy and smoothness of the model. By minimizing V_g, a λ that optimizes the model's fit to the data without overfitting is chosen, making generalized cross-validation especially valuable in scenarios where quick and precise model fitting is essential.

Simulating data

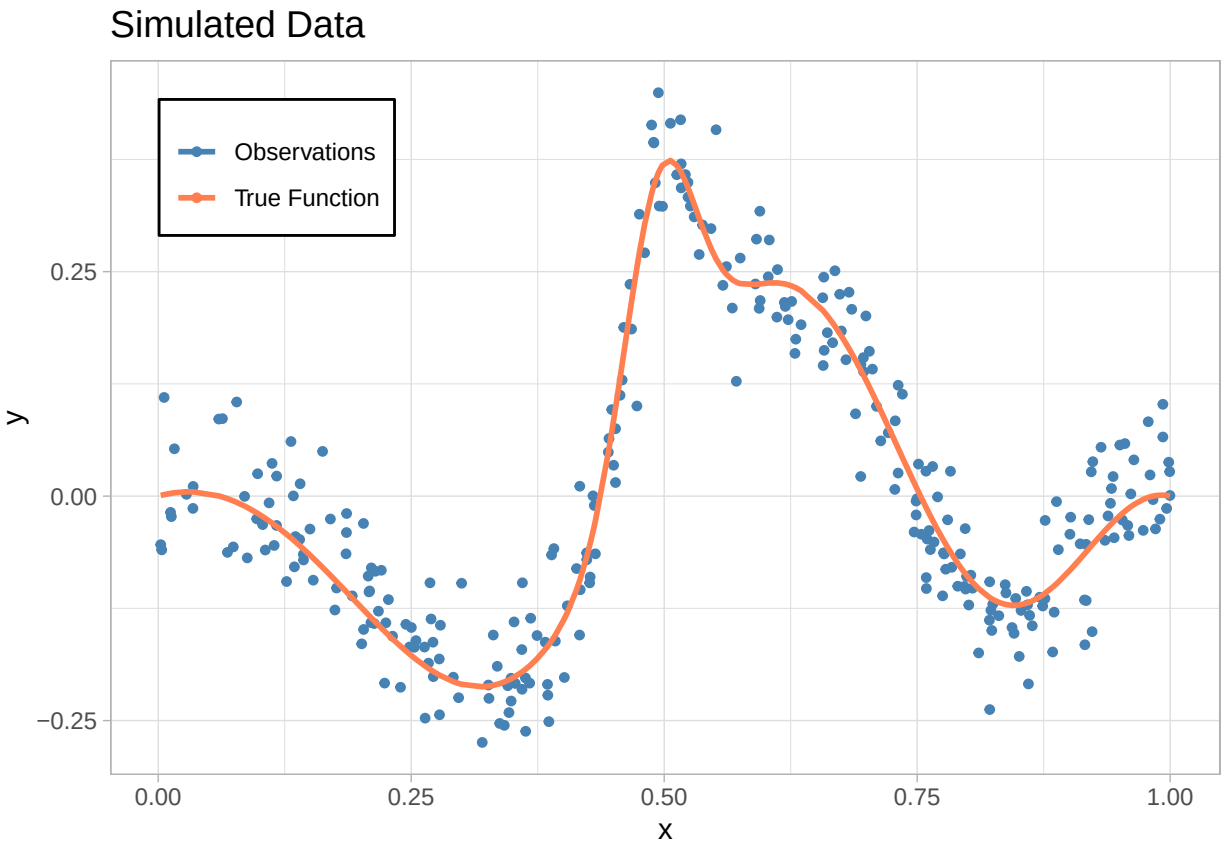
For our examples, we will simulate the data.

```
# Generate X values
x <- runif(300, min=0, max=1)
x <- x[order(x)]

# Error term
eps <- rnorm(300, mean=0, sd=0.05)

# True and observed Y values
y.true <- -x * cos(5 * (x + 0.5)^2) * (x - 1) +
          1.2 * exp(-(20^2) * (x - 0.5)^2) * (x - 1)^2
y.obs <- y.true + eps
```

To visualize the data, we utilize `ggplot2`:

```
library(ggplot2)
```

```
data <- data.frame(x, y.obs, y.true)
ggplot(data = data) +
  geom_point(aes(x = x, y = y.obs, color = "Observations"),
             shape = 20, size = 2) +
  geom_path(aes(x = x, y = y.true, color = "True Function"), linewidth = 1) +
  scale_color_manual(values = c("Observations" = "steelblue",
                                "True Function" = "coral")) +
  labs(title = "Simulated Data",
       x = "x",
       y = "y",
       color = NULL) +
  theme_light() +
  theme(plot.title = element_text(size = 14),
        legend.position = c(0.15, 0.85),
        legend.background = element_rect(fill = "white",colour = "black"))
```



`smooth.spline` Function

To fit smoothing splines, will use the `smooth.spline()` function.

The `smooth.spline()` function fits a smoothing spline that minimizes a weighted combination of the residual sum of squares (RSS) between the curve and the data, and a penalty proportional to the integral of the squared second derivative of the curve. This aims to achieve a curve that is smooth yet fits the data well, controlling the trade-off via the smoothing parameter.

The basic syntax of `smooth.spline()` is as follows:

```
smooth.spline(x, y = NULL, w = NULL, df = NULL, spar = NULL, cv = FALSE, nknots ...)
```

- `x`, `y`: vectors of the coordinates of the data points. `x` are the independent values and `y` are the dependent values.
- `w`: an optional vector of weights for each data point in the fitting.
- `df`: degrees of freedom. Providing this parameter allows control over the smoothness of the spline. If specified, `spar` is ignored.
- `spar`: smoothing parameter. A higher number implies a smoother curve. If `df` is not specified, `spar` can be adjusted to control smoothness.
- `cv`: Logical parameter. Specifies whether to perform cross-validation to select the best smoothing parameter.
  - If `cv = TRUE`: The function performs leave-one-out cross-validation (LOOCV) across the dataset to determine the optimal λ.
  - If `cv = FALSE`: No cross-validation is performed. Instead, the function either:
    - Uses the `spar` parameter if provided to directly set the smoothing level.

    - Uses the `df` parameter to attempt to achieve the specified degrees of freedom by adjusting λ accordingly.
    - Defaults to using generalized cross-validation (GCV) to select λ automatically if neither `spar` nor `df` is specified.
- `nknots`: Specifies the number of knots used to fit the spline.

Fitting the Data

In this section, we will fit a smoothing spline to our data. We will start using cross-validation to automatically determine the optimal smoothing parameter, specifying `nknots=25` to control the number of knots in the spline.

```
smooth_spline_model <- smooth.spline(x, y.obs, cv=TRUE, nknots=25)
```

Next, we display the fitted model object and the selected value of λ that the cross-validation procedure has determined to be optimal:

```
smooth_spline_model
```

```
## Call:
## smooth.spline(x = x, y = y.obs, cv = TRUE, nknots = 25)
##
## Smoothing Parameter  spar= 0.193692  lambda= 1.302541e-05 (14 iterations)
## Equivalent Degrees of Freedom (Df): 20.88272
## Penalized Criterion (RSS): 0.648878
## PRESS(l.o.o. CV): 0.002519592
```

```
smooth_spline_model$lambda
```

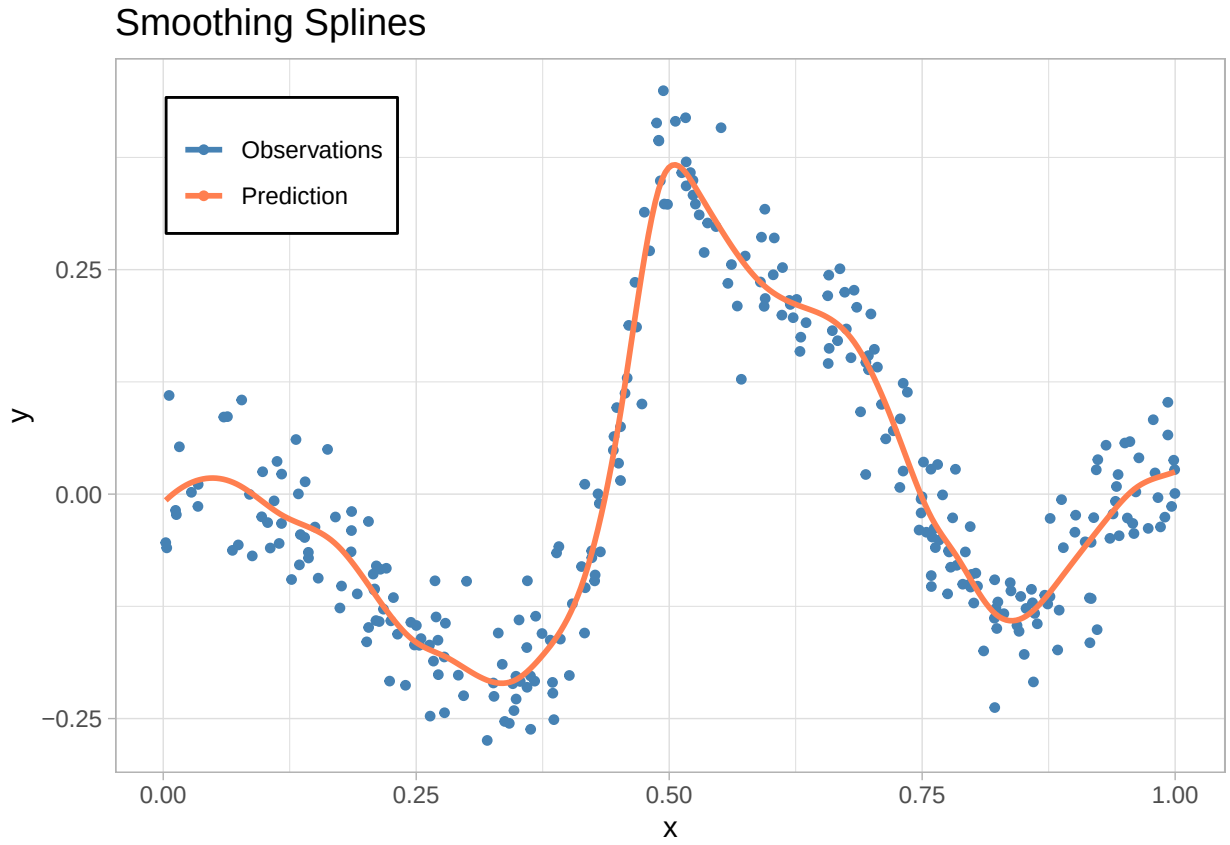
```
## [1] 1.302541e-05
```

Once the model is fitted, we proceed do the predictions.

```
x.pred <- seq(min(x), max(x), length.out = 300)
y.pred <- predict(smooth_spline_model, x.pred)$y

data_spline <- data.frame(x.pred, y = y.pred)
```

We now plot the original data along with the predicted spline to visualize how well the spline fits the data.

```
ggplot(data_spline, aes(x = x)) +
  geom_point(aes(x = x, y = y.obs, color = "Observations"), shape = 20, size = 2) +
  geom_line(aes(x = x.pred, y = y.pred, color = "Prediction"), linewidth = 1) +
  scale_color_manual(values = c("Observations" = "steelblue", "Prediction" = "coral")) +
  labs(title = "Smoothing Splines", x = "x", y = "y", colour=NULL) +
  theme_light() +
  theme(plot.title = element_text(size = 14),
        legend.position = c(0.15, 0.85),
        legend.background = element_rect(fill = "white", colour = "black"))
```




```
# ggsave("../graphs/SmoothingSpline1.pdf", width = 10, height = 6, dpi = 300)
```

To explore how different smoothing parameters affect the fit of a smoothing spline, we will fit the model with three different values of λ. We use `nknots=50` to allow for sufficient flexibility while controlling overfitting.

```
splineS1 <- smooth.spline(x, y.obs, cv=T, lambda=1e-10, nknots=50)
splineS2 <- smooth.spline(x, y.obs, cv=T, lambda=1e-5, nknots=50)
splineS3 <- smooth.spline(x, y.obs, cv=T, lambda=1e-2, nknots=50)
```

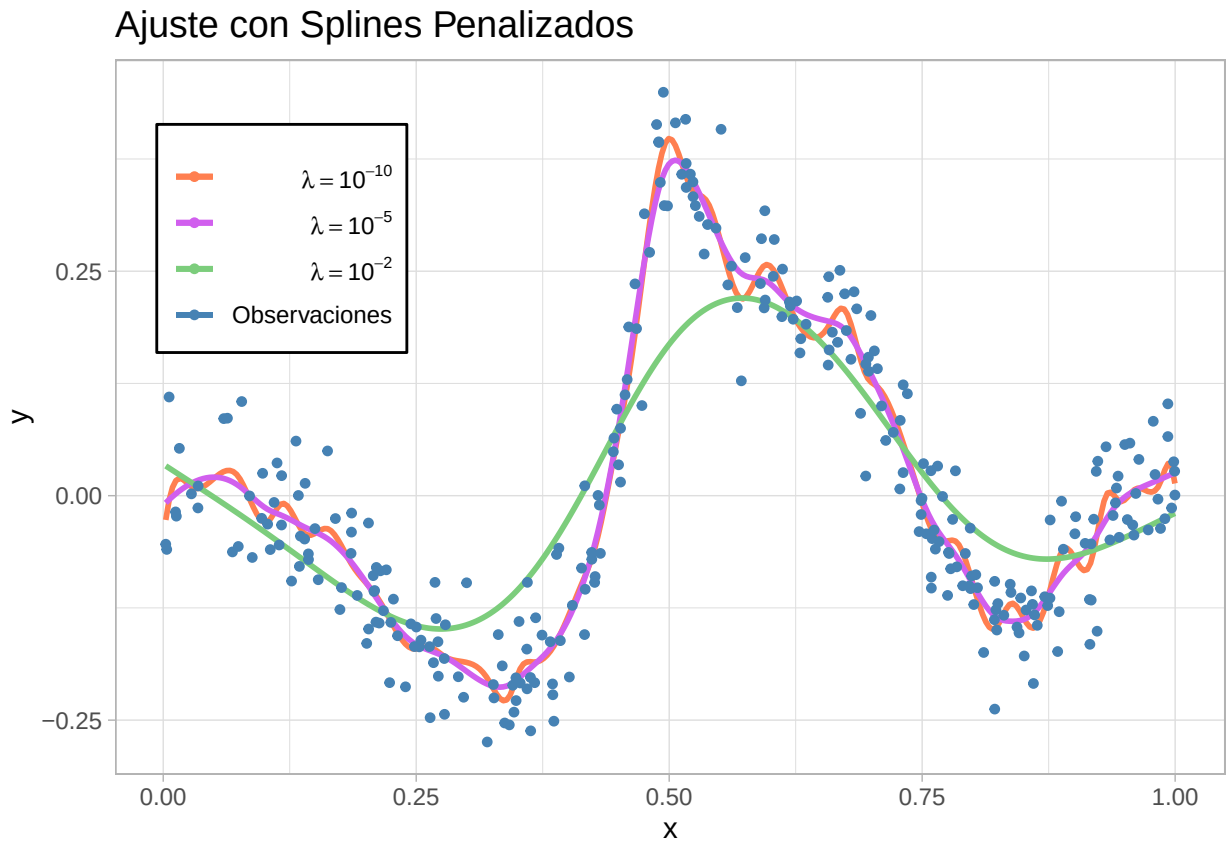
We predict compute the predictions.

```
x.pred <- seq(min(x), max(x), length.out = 300)
y.pred1 <- predict(splineS1, x.pred)$y
y.pred2 <- predict(splineS2, x.pred)$y
y.pred3 <- predict(splineS3, x.pred)$y


data_spline <- data.frame(x.pred, y.pred1, y.pred2, y.pred3)
```

Next, is shown the plot:

```
ggplot(data_spline, aes(x = x)) +
  geom_line(aes(x = x.pred, y = y.pred1, color = "1"), linewidth = 1) +
  geom_line(aes(x = x.pred, y = y.pred2, color = "2"), linewidth = 1) +
  geom_line(aes(x = x.pred, y = y.pred3, color = "3"), linewidth = 1) +
  geom_point(aes(x = x, y = y.obs, color = "Observaciones"), shape = 20, size = 2) +
  scale_color_manual(
    values = c("Observaciones" = "steelblue",
```

```
              "1" = "coral",
              "2"="mediumorchid2",
              "3"="palegreen3"),
  labels = c(expression(lambda == 10^-10),
             expression(lambda == 10^-5),
             expression(lambda == 10^-2),
             "Observaciones")
) +
labs(title = "Ajuste con Splines Penalizados", x = "x", y = "y", colour=NULL) +
theme_light() +
theme(plot.title = element_text(size = 14),
      legend.position = c(0.15, 0.75),
      legend.background = element_rect(fill = "white", colour = "black"))
```




```
# ggsave("../graphs/SmoothingSpline2.pdf", width = 10, height = 6, dpi = 300)
```